Côté, Marie-Pier, Brian Hartman, Olivier Mercier, Josh Meyers, Jared Cummings, and Elijah Harmon. 2025. "Synthesizing Property & Casualty Ratemaking Datasets Using Generative Adversarial Networks." *Variance* 18 (September). https://doi.org/10.66573/001c.144283.

Financial and Statistical Methods

Synthesizing Property & Casualty Ratemaking Datasets using Generative Adversarial Networks

Marie-Pier Côté[1a], Brian Hartman[2b], Olivier Mercier[1c], Josh Meyers[2d], Jared Cummings[2e], Elijah Harmon[2f]

[1] School of Actuarial Science, Laval University, [2] Department of Statistics, Brigham Young University

Keywords: Deep learning, Insurance pricing, Privacy, Tabular data

https://doi.org/10.66573/001c.144283

Variance

Vol. 18, 2025

Due to confidentiality issues, it can be difficult to access or share interesting datasets for methodological development in actuarial science or other fields where personal data are important. We show how to design three different types of generative adversarial networks (GANs) that can build a synthetic insurance dataset from a confidential original dataset. The goal is to obtain synthetic data that no longer contains sensitive information but still has the same structure as the original dataset and retains the multivariate relationships. In order to adequately model the specific characteristics of insurance data, we use GAN architectures adapted for multicategorical data: a Wassertein GAN with gradient penalty (MC-WGAN-GP), a conditional tabular GAN (CTGAN), and a Mixed Numerical and Categorical Differentially Private GAN (MNCDP-GAN). For transparency, the approaches are illustrated using a public dataset, the French motor third-party liability data. We compare the three different GANs on various aspects: ability to reproduce the original data structure and predictive models, privacy, and ease of use. We find that the MC-WGAN-GP synthesizes the best data, the CTGAN is the easiest to use, and the MNCDP-GAN guarantees differential privacy.

This work was supported by an Individual Grant from the Casualty Actuarial Society.

Address for Correspondence: hartman@stat.byu.edu

1. INTRODUCTION

To improve the quality and accuracy of the models used in insurance practice, methodological developments must be tested on the type of data they are meant to model. Unfortunately, insurance claims data at the individual policyholder or claimant level are highly confidential. Just like medical records, these data cannot be publicly shared unless meaningful covariates are erased. The resulting lack of

a Marie-Pier Côté is an associate professor at the School of Actuarial Science at Laval University in Québec, Canada. Prior to joining Laval, she completed a master's and a PhD in statistics at McGill University under the supervision of Christian Genest. She is also a fellow of the Society of Actuaries and an associate of the Canadian Institute of Actuaries. Her research interests lie in statistical methods, algorithmic fairness, and copula models for general insurance applications.

b Brian Hartman is a professor and actuarial program coordinator at Brigham Young University. He has worked in various capacities with companies in property-casualty, health, and long-term care insurance. He holds a bachelor's degree in actuarial science from Brigham Young University and a PhD in statistics from Texas A&M University. He is also an associate of the Society of Actuaries. Before BYU, he was a postdoctoral fellow at the University of Waterloo (2010–2011) and an assistant professor at the University of Connecticut (2011–2015).

c Olivier Mercier has a master's degree in computer science–artificial intelligence from Laval University in Québec. He is currently working for Co-Operators, an insurance company, as part of a data science team. His work focuses on solving a wide array of business problems through the exploration and analysis of data. He is interested in the practical application of statistical and machine learning techniques and models.

d Josh Meyers is currently working as a data scientist at a property and casualty insurance company. He holds a bachelor's degree in actuarial science and a master's degree in statistics, both from Brigham Young University. As a data scientist, he utilizes machine learning techniques to create innovative solutions to traditional actuarial problems and processes.

e Jared Cummings holds a master's degree in statistics from Brigham Young University. He currently works as a data scientist at USAA Federal Savings Bank, where he uses statistical learning to detect and prevent financial crimes.

f Elijah Harmon is an actuarial associate at Voya Financial, with a bachelor's degree in actuarial science from Brigham Young University.

publicly available data slows down methodological developments in actuarial science.

Loss reserving methods provide an example. With the improved availability of computing resources, reserving methods that traditionally used aggregate information may now model individual claims. Antonio and Plat (2014), Pigeon, Antonio, and Denuit (2013, 2014), and Wüthrich (2018a) proposed micro-level reserving models and illustrated their efficacy on confidential datasets. Because the data are confidential, it is difficult to compare the methods across authors or to new methods yet to be developed. Additionally, the research is not easily reproducible, even when the code is shared.

Gabrielli and Wüthrich (2018) discussed this lack of publicly available data and provided an R program for simulating insurance claim development patterns. A Gaussian copula with appropriate margins generates the features, and the different parts of the development process are modeled with successive neural nets. The simulation machine accommodates only a few covariates; therefore, generating a large number of features with the Gaussian copula could lead to unrealistic combinations of factor levels. In this paper, we propose synthesizing insurance data with a generative adversarial network (GAN).

A GAN is a deep-learning model introduced by Goodfellow et al. (2014). It consists of two competing neural networks: a *generator* that generates fake data and a *discriminator* that is trained to identify whether the data are real or fake. During the training process, the generator adapts to fool the discriminator, which means that it learns to generate fake data that are indistinguishable from the real data. The resulting GAN could thus be used to simulate a synthetic dataset that is completely fake but still has the structure of real data.

Frid-Adar et al. (2018) used GANs to generate synthetic data to augment a small imaging dataset and improve the performance of liver lesion classification. As Papernot et al. (2017) explained, a method based on GANs can provide strong privacy for sensitive training data. Choi et al. (2017) proposed the medGAN architecture to synthesize realistic patient records. Their motivation was similar to ours in that patient records are highly confidential but extremely valuable for developing new models and statistical methods. The structure of patient record data is also closer to that of insurance data, as compared with the data used in most of the deep-learning GAN literature, which focuses on unstructured data such as images. Images (and pixels) are continuous, whereas most claimant characteristics are categorical variables. This adds complexity because one cannot interpolate between discrete classes to create fake records. Camino, Hammerschmidt, and State (2018) adapted the medGAN and the Wasserstein GAN with gradient penalty (WGAN-GP) from Gulrajani et al. (2017) for multicategorical variables.

Deep learning has received increasing attention in recent actuarial research. Schelldorfer and Wüthrich (2019) applied a generalized linear model embedded in a neural network to analyze the French motor third-party liability claims dataset (studied in Section 5). Wüthrich (2018b) used neural networks for chain-ladder reserving. However, to the best of our knowledge, only Kuo (2019) has used a type of GAN in actuarial applications to date.

In this paper, we introduce other GANs to the actuarial science literature and adapt the metrics to be appropriate for Poisson count data. Although some frequency datasets are publicly available to develop and test pricing methods, they are toy datasets compared with those that are kept confidential, because they contain few policyholders or lack complex covariates, such as telematics or spatial information. We present and test three architectures. The first, in Section 2, is based on Camino et al.'s (2018) multicategorical adaptation of the WGAN-GP. Section 3 presents the conditional tabular GAN from L. Xu et al. (2019), applied to ratemaking data in Kuo (2019). We call the last model, detailed in Section 4, the mixed numerical and categorical differentially private GAN, or MNCDP-GAN. This model is an adaptation of the differentially private GAN with autoencoder developed by Tantipongpipat et al. (2019). The MNCDP-GAN is the only model that incorporates differential privacy, which is the gold standard for guaranteeing that data can be shared without confidentiality issues. In Section 5, we test in a case study the three architectures using the French motor third-party liability dataset, publicly available in the R package `CASdatasets` (Dutang and Charpentier 2019). All code is available in the GitHub repository for this paper.[1] Section 6 concludes the paper and is followed by Appendices A and B, which detail the setup and tuning of the multicategorical and continuous WGAN-GP and MNCDP-GAN, respectively.

2. MULTICATEGORICAL WASSERSTEIN GAN

Let us first introduce the general framework of GANs. GAN training is a game between two competing networks, the generator and the discriminator. The generator G is a neural net with parameter vector θ_g that takes in argument a vector of random noise Z with distribution F_z and maps it to the space of the data we wish to model. Usually, the components of vector Z are independent standard Gaussian random variables and the dimension of Z is lower than that of the data. The resulting $G(Z;\theta_g)$ is a fake data point, and its distribution is denoted by F_g.

The goal of the training procedure is therefore to find a good approximation F_g of the unknown distribution of a true data point X, denoted F_x. To achieve this goal, a competing network, the discriminator D with parameter vector θ_d, learns to determine whether a data point is real or fake. To this end, the parameters θ_d of D are trained to maximize the expected score of a real data point $\mathrm{E}_X\{D(X;\theta_d)\}$ and

1 https://github.com/brianmhartman/Anonymizing-Ratemaking-Datasets-using-GANs

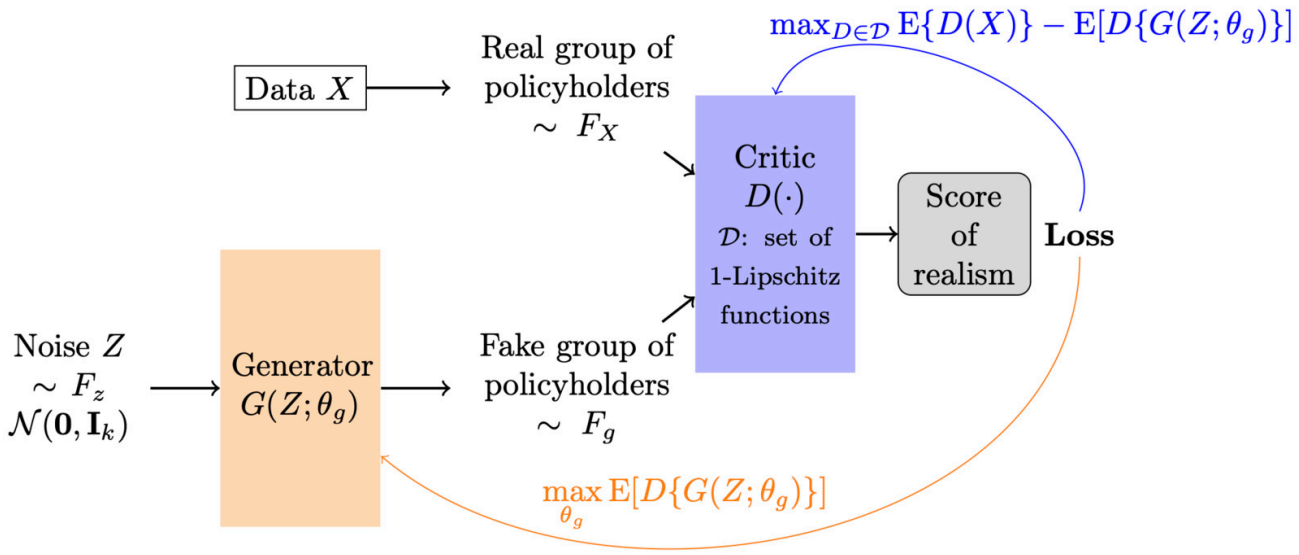


Figure 1. WGAN schema. The arrows represent the flow of the training process.

to minimize the expected score of a synthetic data point $\mathrm{E}_Z[D\{G(Z;\theta_g);\theta_d\}]$. To achieve the goal of generating realistic data points, the parameters θ_g of the generator are trained to maximize the discriminator's score on a *fake* data point $\mathrm{E}_Z[D\{G(Z;\theta_g);\theta_d\}]$. Combining the two problems, the two networks aim to solve

$$\min_{\theta_g}\max_{\theta_d} \mathrm{E}_X[\log\{D(X;\theta_d)\}] + \mathrm{E}_Z(\log[1 - D\{G(Z;\theta_g);\theta_d\}]).$$

This optimization problem minimizes the Jensen-Shannon divergence between F_x and F_g. In practice, this leads to serious convergence issues, partly solved by training D and G in turn with minibatches.

To solve some of the convergence issues, Arjovsky, Chintala, and Bottou (2017) advocated using the Wasserstein-1 distance between F_x and F_g; that is, they considered the problem

$$\min_{\theta_g}\max_{D\in\mathcal{D}} \mathrm{E}_X\{D(X)\} - \mathrm{E}_Z[D\{G(Z;\theta_g)\}], \qquad (1)$$

where $\mathcal{D}$ is the set of 1-Lipschitz functions. This change in the objective function leads to the Wasserstein GAN, or WGAN. The discriminator in a WGAN is called the *critic*, as it outputs a real value rather than a binary classification. The WGAN is depicted schematically in Figure 1 for policyholder claim data X. The black arrows represent the forward flow of information in the network, while the colored arrows represent the flow of the training process for the generator (orange) and critic (blue).

Some tactics are needed to enforce the Lipschitz constraints on D. In this regard, the gradient penalty (GP) developed by Gulrajani et al. (2017) greatly improves the WGAN training. In their WGAN-GP model, the authors take advantage of the fact that a differentiable Lipschitz function has gradients with norm at most 1 everywhere. A tuning parameter $\lambda > 0$ is introduced, and the objective of the WGAN-GP is

$$\begin{aligned}\min_{\theta_g}\max_{\theta_d} &\mathrm{E}_X\{D(X;\theta_d)\} \\ &-\mathrm{E}_Z[D\{G(Z;\theta_g);\theta_d\}] \\ &+\lambda\mathrm{E}_{\hat{X}}[\{||\nabla_{\hat{x}} D(\hat{X};\theta_d)||_2 - 1\}^2],\end{aligned}$$

where $\hat{X} \overset{d}{=} UX + (1-U)G(Z;\theta_g)$, and U is uniformly distributed on the interval $(0,1)$, so that the distribution $F_{\hat{x}}$ of $\hat{X}$ is obtained by sampling uniformly along lines between pairs of points sampled from F_x and F_g. For details on the motivation, the reader is referred to Gulrajani et al. (2017).

In practice, if $m \in \mathbb{N}$ is the size of the minibatch with observations $x_1, \ldots, x_m$, random noise vectors $z_1, \ldots, z_m$, and independent uniform samples $u_1, \ldots, u_m$, then we let $\hat{x}_i = u_i x_i + (1-u_i)G(z_i;\theta_g)$ and the discriminator loss is approximated by

$$\begin{aligned}\mathcal{L}_d = \frac{1}{m}\sum_{i=1}^{m} &-D(x_i,\theta_d) + D\{G(z_i;\theta_g);\theta_d\} \\ &+\lambda\{||\nabla_{\hat{x}_i} D(\hat{x}_i;\theta_d)||_2 - 1\}^2,\end{aligned}$$

while the generator loss is simply

$$\mathcal{L}_g = \frac{1}{m}\sum_{i=1}^{m} -D\{G(z_i;\theta_g);\theta_d\}.$$

Note that higher values of the critic D indicate fake samples.

The WGAN and WGAN-GP were developed in the context of image generation tasks. However, in the current application we wish to synthesize tabular insurance data, in which some variables are categorical with multiple levels. Camino, Hammerschmidt, and State (2018) considered an application close to ours where the target data contain many multicategorical variables. They modified the WGAN-GP generator so that, after the model output, there is a dense layer in parallel for each categorical variable followed by a softmax activation function. Then, the results are concatenated to yield the final generator output.

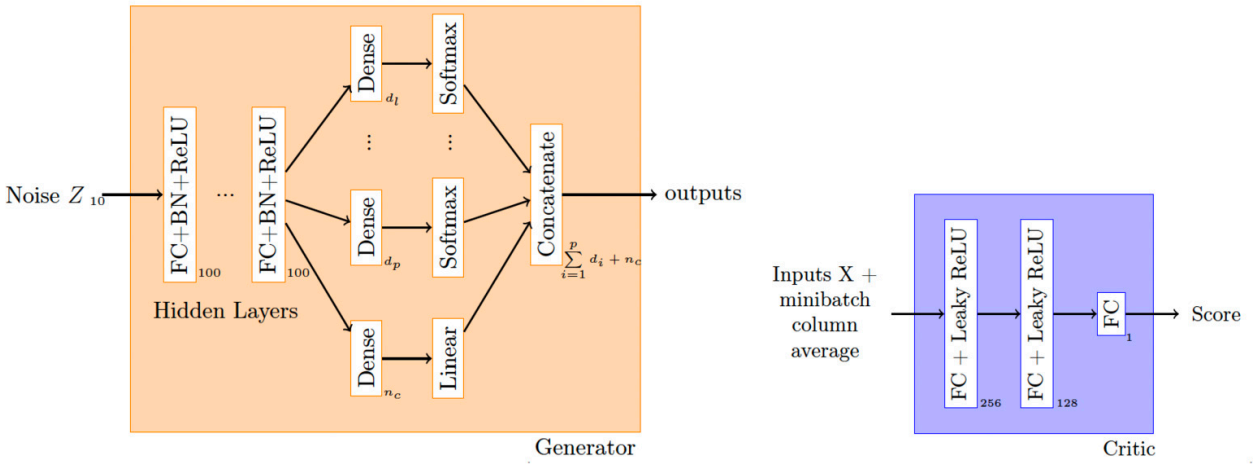


Figure 2. Architecture inside the generator (orange) and the critic (blue) for our multicategorical and continuous WGAN. The dimensions $d_1, \ldots, d_p$ represent the number of levels in categorical variables $1, \ldots, p$. FC stands for fully connected and BN stands for batch normalization.

As in Camino, Hammerschmidt, and State (2018), our generator's architecture has one dense layer with dimension matching the number of levels for each multicategorical variable. We also add one dense layer with linear activation and dimension n_c, which is equal to the number of continuous variables. The architecture of the generator and critic in our multicategorical and continuous WGAN-GP, or MC-WGAN-GP, is depicted in Figure 2. Further details about hyperparameter optimization are available in Appendix A.

3. CTGAN

Another possible path to simulating insurance claim data is through a conditional tabular GAN or CTGAN (L. Xu et al. 2019). This method was applied to ratemaking data in Kuo (2019). Additionally, Kuo developed an R wrapper for this software to make it easily accessible to insurance practitioners more familiar with R than Python. Starting from his code, we slightly adjusted the preamble to improve the application consistency on our machines and slightly adjusted the preprocessing, but other than that the overall code remained the same. Our version of the code is available in the GitHub repository for this paper.

The CTGAN simulates records one by one. It first randomly selects one of the variables (say fuel type: diesel or gasoline). Then, it randomly selects a value for that variable (say diesel). Following Kuo (2019), we use the true data frequency to sample the value rather than the log-frequency, as suggested in L. Xu et al. (2019). Given that value for that variable, the algorithm finds a matching row from the training data (in this example, it randomly selects a true observation with a diesel-powered car). It also generates the rest of the variables conditioning on it being diesel-powered. The generated and true rows are sent to the critic, which gives a score. Figure 3 summarizes the CTGAN procedure.

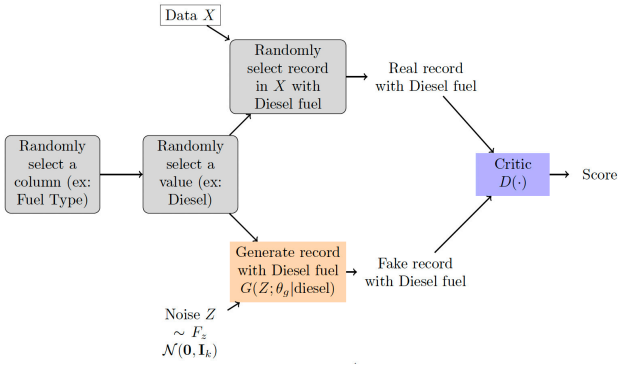


Figure 3. CTGAN schema. The model flow is illustrated for the case when the selected column is Fuel Type and the selected value for that column is Diesel.

Figure 4 zooms inside the architecture of the generator and critic. Both the critic (blue) and generator (orange) use two fully connected layers to attempt to capture all relationships between the columns. An additional sophistication of the CTGAN is the use of the PacGAN framework (Lin et al. 2018) in the discriminator, where 10 samples are provided in each pac to prevent the mode collapse issue. As in D. Xu et al. (2018), the model is trained using the WGAN-GP loss.

Like the previously discussed MC-WGAN-GP, the CTGAN does not incorporate privacy protections, though that could possibly be developed, as hinted in Kuo (2019).

4. MNCDP-GAN

The mixed numerical and categorical differentially private GAN (MNCDP-GAN) tries to solve the drawbacks of the other two GANs. The MNCDP-GAN includes an autoencoder and a WGAN. The main advantage of this architecture, introduced in Tantipongpipat et al. (2019), is that

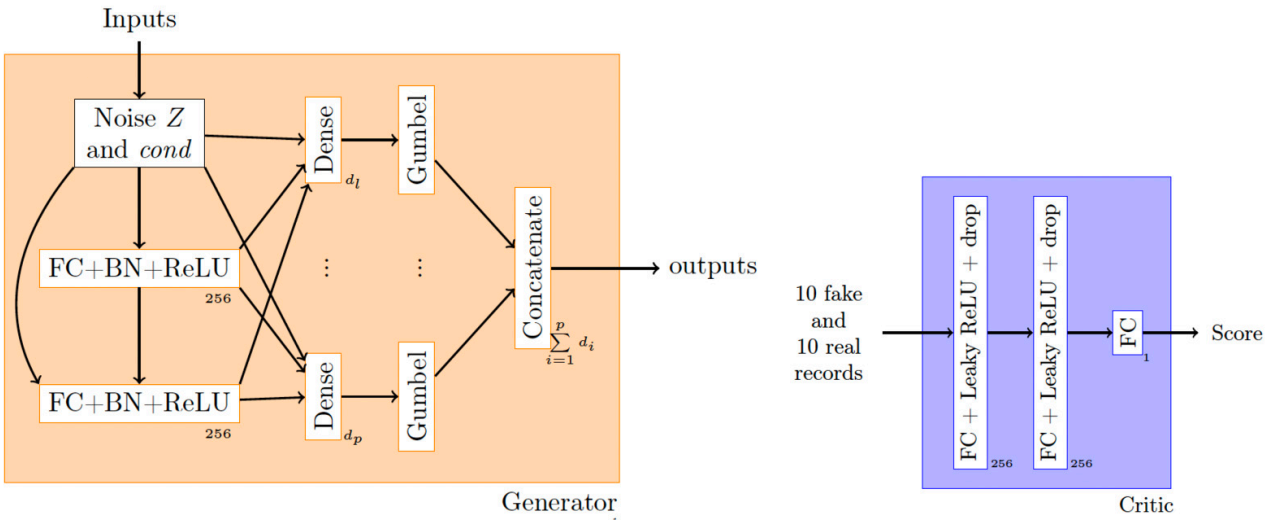


Figure 4. Architecture inside the CTGAN generator (orange) and critic (blue). The dimensions $d_1, \ldots, d_p$ represent the number of levels in categorical variables $1, \ldots, p$. The input of the generator is Gaussian random noise and the condition *cond* of the feature value that was randomly selected (see Figure 3). FC stands for fully connected, BN for batch normalization, Gumbel for the Gumbel softmax activation, and drop for dropout.

the generator works in a latent space of encoded variables, which can be easier to model adequately than the original structured data. Training can be done in a differentially private (DP) manner, allowing a DP guarantee on the generated dataset.

As depicted in Figure 5, the original data are first preprocessed (one-hot encodings for categorical variables and either binning or min-max standardization for continuous variables), resulting in vectors defined in $[0,1]^n$ that are fed into an encoder, shrinking the dimension to $d < n$, a hyperparameter. Then, a decoder enters the encoded variable in the latent space $\mathbb{R}^d$ and outputs data in the format $[0,1]^n$, which is subsequently postprocessed to deliver data in the original format. This architecture is called an autoencoder and is used in many neural network applications. In our context, the autoencoder creates the latent space in dimension d, which is easier for the generator to learn because it has less structure than the original data space. The generator takes in random noise and outputs a vector in the latent space, which can then be decoded by the decoder to produce a synthesized record. The critic is trained with the Wasserstein loss and compares the generated data before postprocessing with the preprocessed original data.

In Figure 5, the autoencoder flow and training are depicted in red, the data flow in the autoencoder and critic is indicated in green, and the generated data flow through the decoder and the critic is highlighted in orange. It is reasonably assumed that the postprocessing step can be done using public knowledge and does not affect the model's DP quality. The DP training is done by injecting noise in the decoder and critic. For more details, refer to Tantipongpipat et al. (2019).

The level of differential privacy achieved by the model (including both the autoencoder and the GAN) is quantified by the value $\epsilon > 0$. This value relates to how different an analysis may be if one data point is added or removed. If the dataset X' is identical to X except for one added data point, then an (ϵ, δ) differentially private analysis $\mathcal{M}$ satisfies

$$\Pr\{\mathcal{M}(X) \in S\} \leq e^{\epsilon} \Pr\{\mathcal{M}(X') \in S\} + \delta,$$

for $\delta > 0$, any such X, X' and $S \subseteq Range(\mathcal{M})$, as seen, for example, in Dwork and Roth (2014). A smaller ϵ represents stronger privacy guarantees, but comes with decreased performance for synthesizing realistic data because more noise is added to the training. The values of ϵ and δ for our procedure are obtained through a *privacy accountant* as explained in Tantipongpipat et al. (2019). Further details on our implementation are available in Appendix B.

5. CASE STUDY

To show the value of the three approaches in a reproducible manner and compare their effectiveness in producing synthetic data, we use a well-known publicly available dataset for the case study. The dataset contains a set of 412,748 French motor third-party liability policies observed in a single year (Dutang and Charpentier 2019). The data contain the number of claims (ClaimNb), along with eight explanatory variables:

- Exposure: the number of car-years on the policy, bounded between 0 and 1 (we removed the few records with Exposure greater than 1)
- Power: an ordered categorical variable that describes the power of the vehicle
- CarAge: the vehicle age in years
- DriverAge: the age of the primary driver, in years
- Brand: the vehicle brand divided into the following groups: A— Renault, Nissan, and Citroen; B— Volkswagen, Audi, Skoda, and Seat; C— Opel, General Motors, and Ford; D— Fiat, E— Mercedes, Chrysler, and BMW; F— Japanese (except Nissan) and Korean; G— other

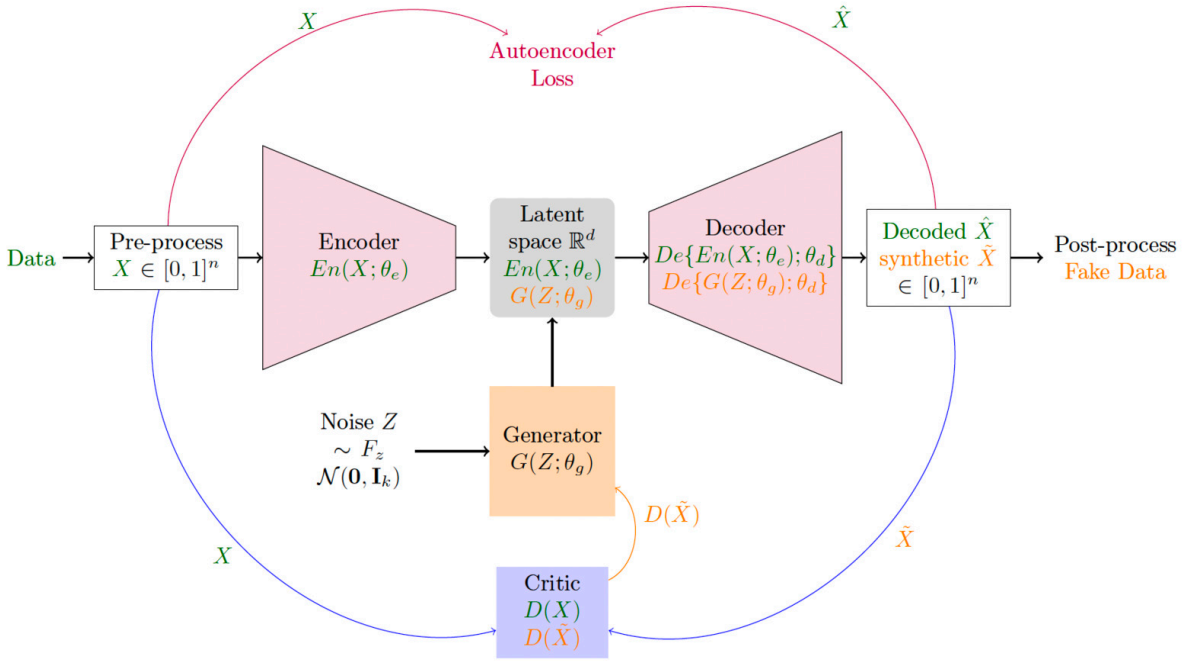


Figure 5. MNCDP-GAN schema. The orange relates to the generated data, and the green relates to the original data. The colored arrows represent the flow into the loss for the training of each network, with autoencoder in red, generator in orange, and discriminator in blue. For DP training, noise is added in training the decoder and the critic.

- Gas: diesel or regular
- Region: the policy region in France
- Density: number of inhabitants per km^2 in the home city of the driver

Brand, Gas, Power, and Region are all categorical variables and the other four are numeric (continuous or discrete). For the MNCDP models we show four DP levels:

- $\epsilon = \infty$ is labeled on the plots as "MNCDPInfty": no differential privacy
- $\epsilon = 100,000$ labeled as "MNCDP100k"
- $\epsilon = 10,000$ labeled as "MNCDP10k"
- $\epsilon = 5$ is labeled as "MNCDP5": strong differential privacy

We simulate a dataset of the same size as the original dataset using each of the GANs and compare the univariate distributions in the generated samples with the univariate distributions in the real data. If the methods faithfully reproduce the original data, we expect the distributions to be similar.

We first compare the results for the categorical variables. Figure 6 shows the observations in each category in the real and generated datasets for Brand, Gas, Power, and Region for the real data, the MC-WGAN-GP, the CTGAN, and the MNCDP-GAN without DP. Figure 7 provides the same information on the number of claims. From the univariate perspective, the three models all replicate the real data reasonably well. In particular, the MC-WGAN-GP (green) closely reproduces the univariate distributions in the real data (red) for these four categorical variables and the response variable.

Figure 8 shows the same information as Figure 6, but for the MNCDP model with varying levels of differential privacy. It is readily apparent that the quality of synthesized data declines markedly as the level of differential privacy increases. Again, the model with $\epsilon = \infty$ (MNCDPInfty), no differential privacy, follows the data rather well. As ϵ decreases to 100,000 (MNCDP100k), the model still approximates the real data relatively well. But the synthetic data when $\epsilon = 10,000$ (MNCDP10k) and, especially, $\epsilon = 5$ (MNCDP5) are not close to the real data. The noise added to the process in both cases completely obscures the original signal. This is a consistent result throughout our case study.

Shown another way, Figure 9 plots the frequency for each category in the real data on the x-axis against the frequency in the synthesized data on the y-axis for each of the GANs we considered, color-coded by feature. The line $y = x$ is also plotted. The MC-WGAN-GP dataset seems to match the real frequencies best, followed by the CTGAN, MNCDPInfty, and MNCDP100k models (which are relatively similar). As noted above, the MNCDP10k and MNCDP5 models drastically differ from the original data.

For the numeric variables CarAge, Density, DriverAge, and Exposure, Figure 10 shows the distributions of the real data in the top row and compares them with the distributions of data generated by the models. The Exposure variable is one of the most difficult aspects of insurance data because a large proportion of Exposure values are exactly one. After accounting for those, the remaining Exposure

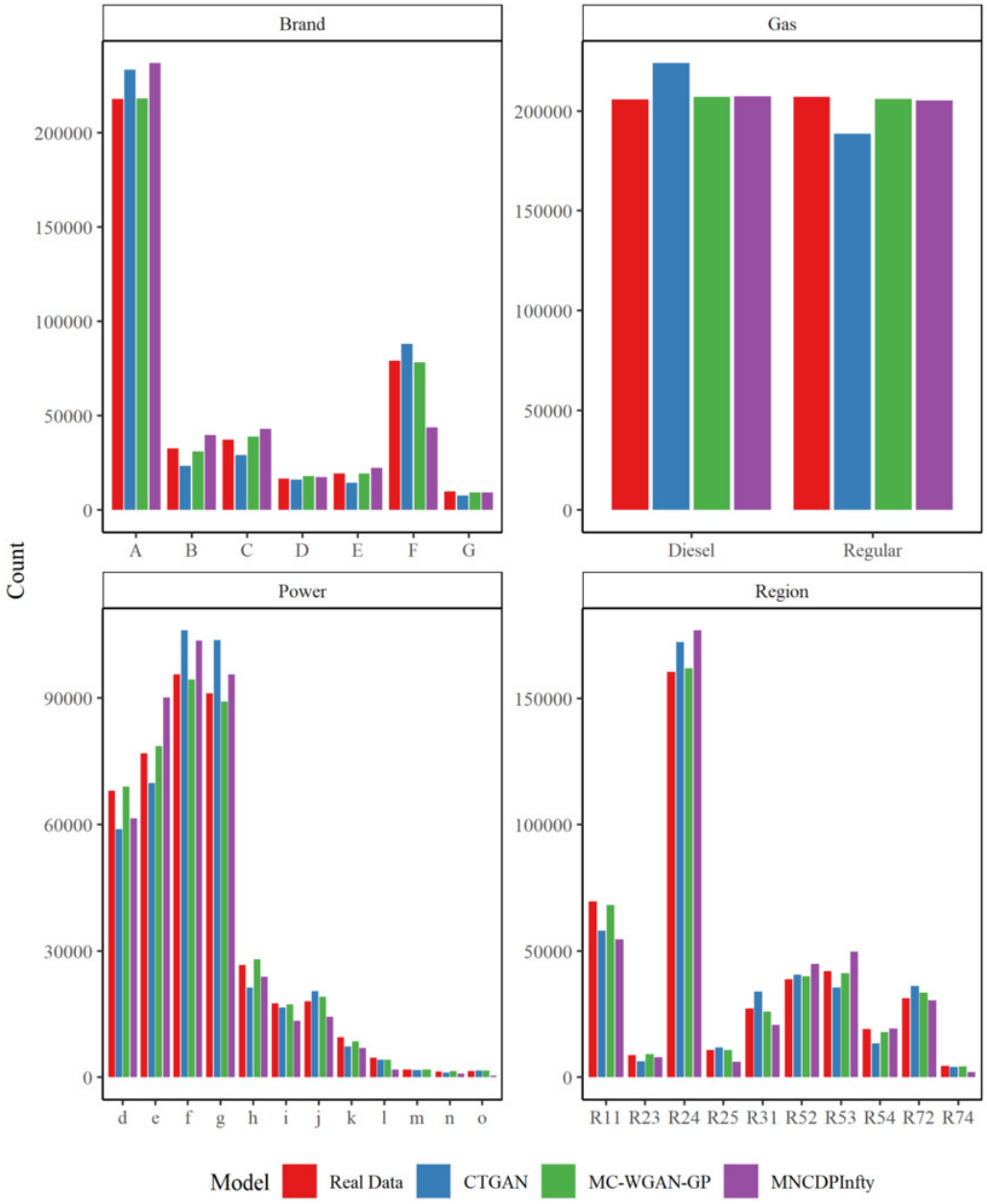


Figure 6. Comparison of univariate categorical variable distributions.

values tend to be either close to zero or close to monthly intervals (1/12, 2/12, 3/12, etc.). Both the CTGAN and MC-WGAN-GP do well with synthesizing the correct number of 1 values, but in the rest of the distribution, the MC-WGAN-GP is too bumpy and the CTGAN might be too smooth. CTGAN is the best model for the Density variable. Both DriverAge and CarAge are matched well by all three methods.

It is important to correctly model the univariate characteristics, but it is even more important to correctly model the multivariate relationships. This is especially true with the relationship between claim counts and the various explanatory variables. Figure 11 compares the probability of a claim in each categorical group. The real probability of a claim is on the x-axis, with the synthesized probability on the y-axis. The line $y = x$ is also plotted to show the ideal goal. The size of the marks shows the proportion of synthesized data in each group. By this metric, the MC-WGAN-GP performs the best, followed closely by the CTGAN. The MNCDP-GAN does not perform well, even without differential privacy.

Our last test examines the consistency of models fitted on the original and synthesized data. We split the real data into two parts, a 70% training set and a 30% test set. We fit a Poisson generalized linear model on the training set, predicting the number of claims using all eight explanatory variables. We then fit the same model on a 70% sample from each of the synthesized datasets. The sampling and model fitting are performed 5,000 times to examine the sampling variability and to obtain more consistent estimates. Figure 12 compares the average estimated regression coefficients for each of the three models. We find that the CTGAN coefficients are all close to the real coefficients.

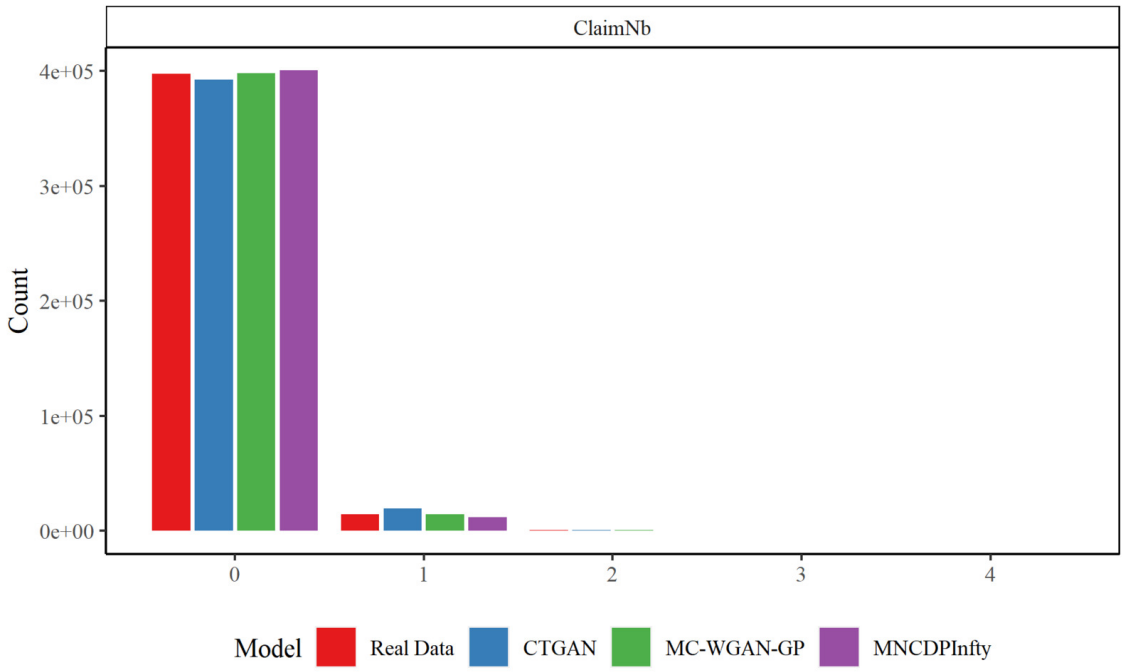


Figure 7. Comparison of response variable distributions.

The coefficients estimated with the MC-WGAN-GP data are similarly close, except for a single region coefficient. The results achieved using the MNCDP synthesized data are again the worst.

With each fitted model, we then predict the claim counts for the 30% real test data and compare the predictions from the models fit on the synthesized data to the predictions from the models fit on the real data. Table 1 shows the median absolute error (MAE) and mean squared error (MSE) between the predictions, with 95% bootstrapped confidence intervals. The MC-WGAN-GP significantly outperforms the other two models on both metrics. The CTGAN performs next best, and the MNCDP-GAN again shows the worst performance.

6. CONCLUSION

In this paper, we presented, implemented, and compared three methods to synthesize insurance data. All three methods were based on GANs, and each method had advantages and disadvantages. The MC-WGAN-GP method synthesized the most realistic data, generating data that were very similar (accounting for both univariate and multivariate relationships) to the real data. The CTGAN method was the easiest to use, especially for individuals more familiar with R than Python. The data synthesized by CTGAN were almost as good as the MC-WGAN-GP data. The main drawbacks of MC-WGAN-GP and CTGAN were that they provided no privacy guarantees; some records in the generated data could still contain confidential information. The MNCDP-GAN incorporated differential privacy, but its synthesized data (even without differential privacy) were not as good as those produced by the other two methods.

Future work can start from any of the three models and attempt to add the advantages of the other two—that is, either add differential privacy and ease of use to the MC-WGAN-GP, add improved synthesis and differential privacy to the CTGAN, or add improved synthesis and ease of use to the MNCDP-GAN. In any case, GANs are a promising tool for synthesizing and protecting private data that are important to actuarial science and other fields.

ACKNOWLEDGMENTS

This work was supported by a Casualty Actuarial Society (CAS) Individual Grant and by M.-P. Côté's Chair in Educational Leadership in Big Data Analytics for Actuarial Sciences — Intact. The authors thank the members of the CAS project oversight committee, Syed Danish Ali, Morgan Bugbee, Marco De Virgilis, and Greg Frankowiak, for useful feedback throughout the project. The project would not have been possible without the computing resources provided by the Digital Research Alliance of Canada, and the statistics department computing cluster at Brigham Young University.

Submitted: August 14, 2020 EDT. Accepted: January 21, 2022 EDT. Published: September 29, 2025 EDT.

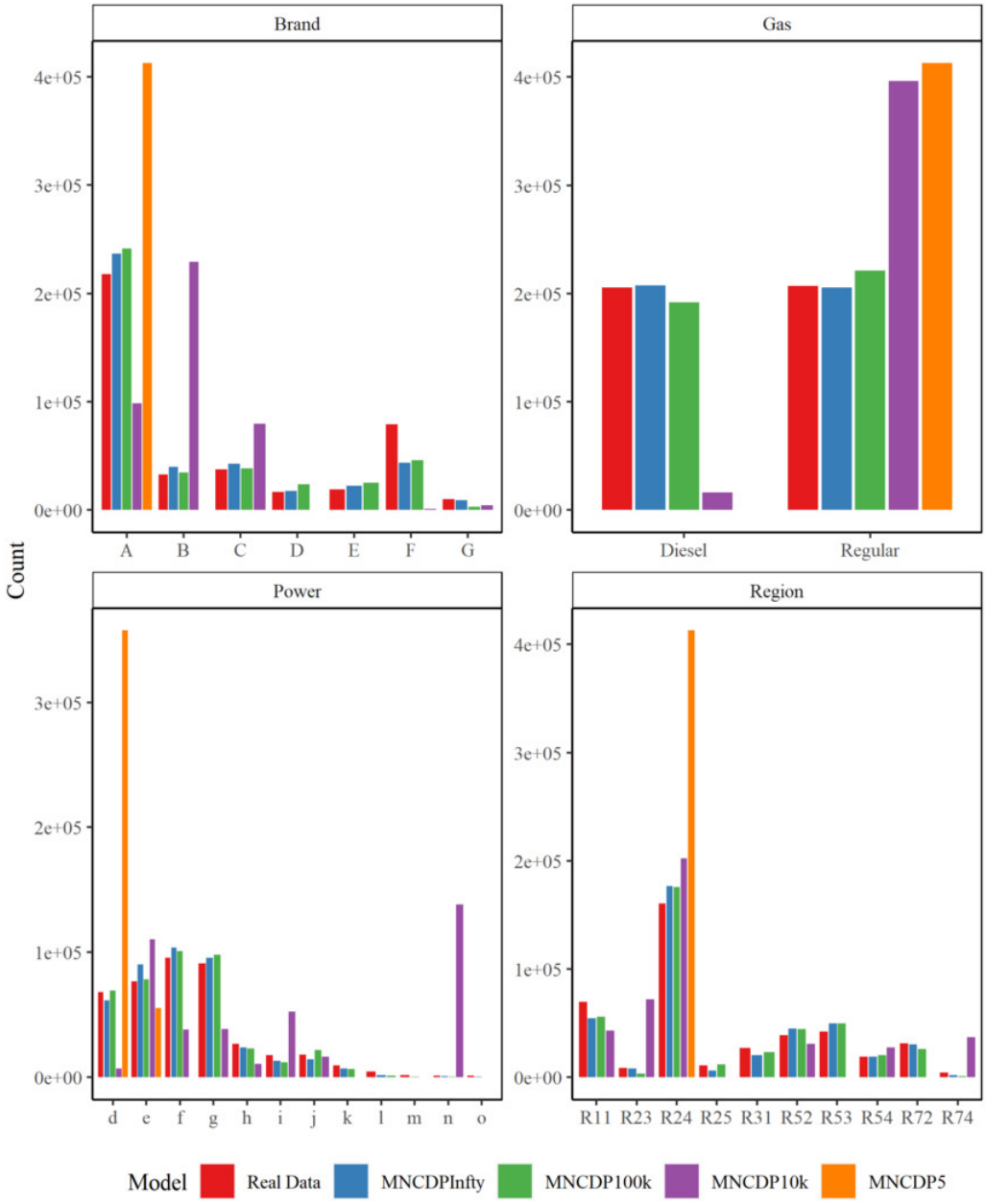


Figure 8. Comparison of univariate categorical variable distributions for MNCDP-GAN models with $\epsilon \in \{5, 10000, 100000, \infty\}$.

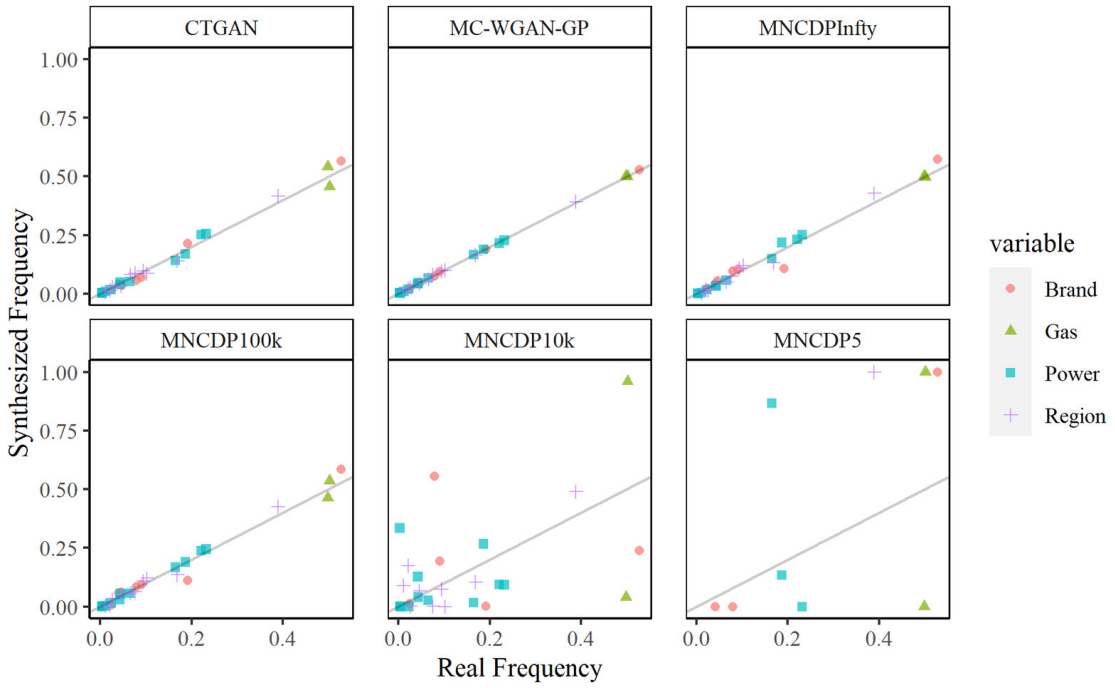


Figure 9. Comparison of the synthesized and real group frequencies for each class of the four categorical variables.

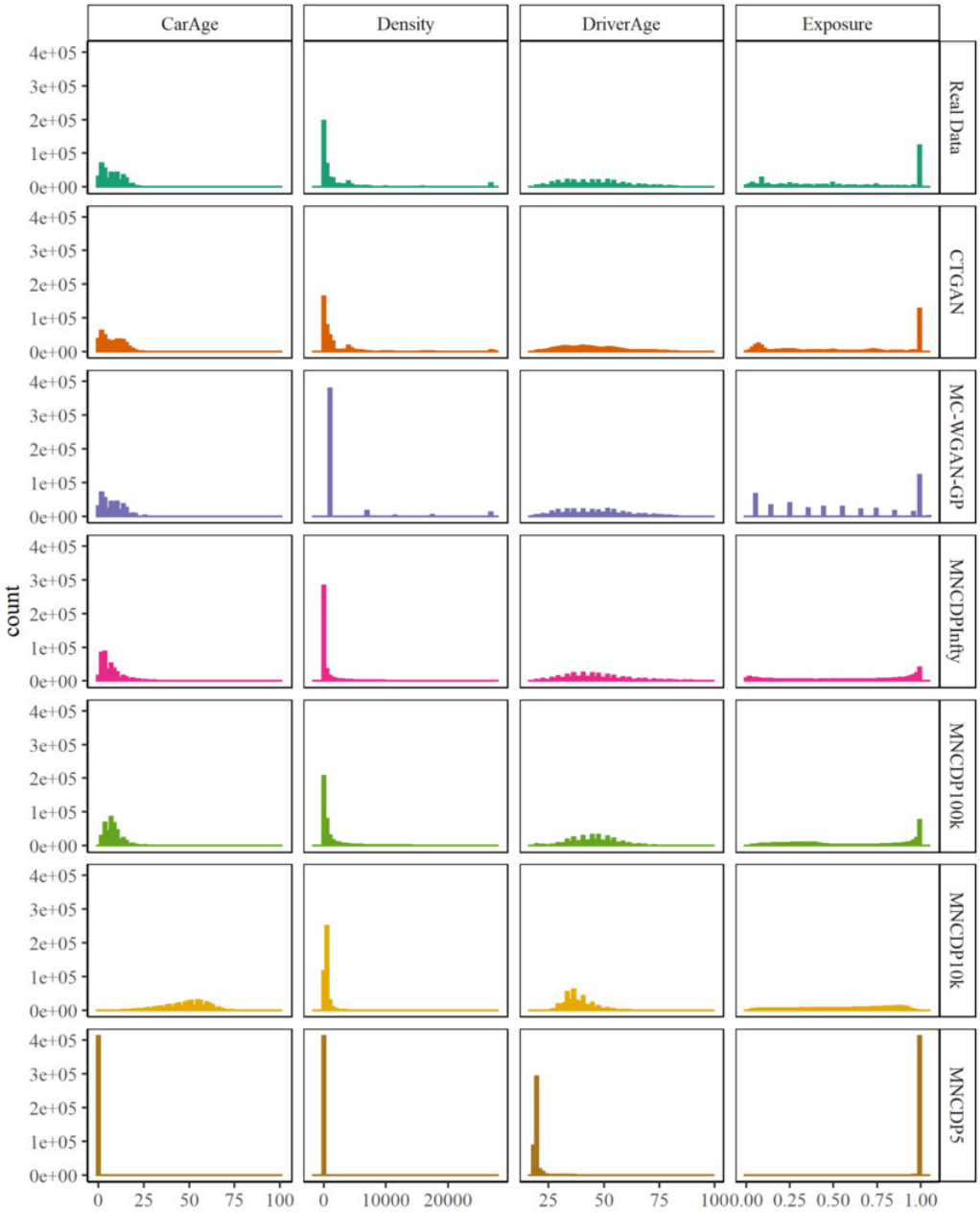


Figure 10. Comparison of univariate numerical variable distributions.

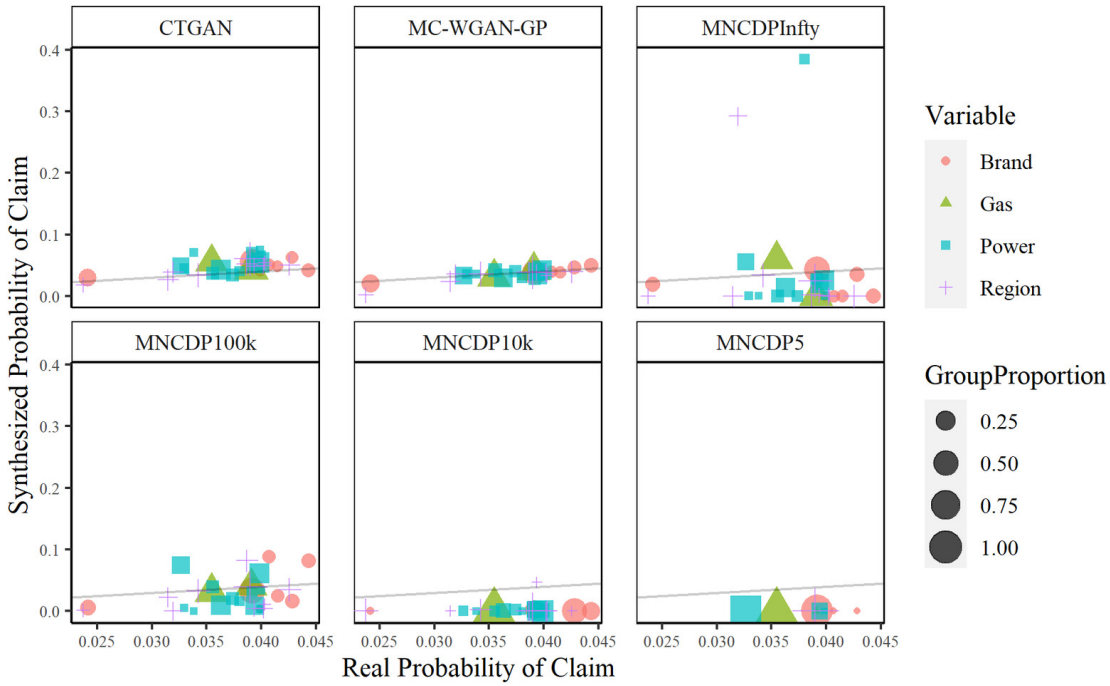


Figure 11. Comparison of the synthesized and real empirical probability of a claim for each group.

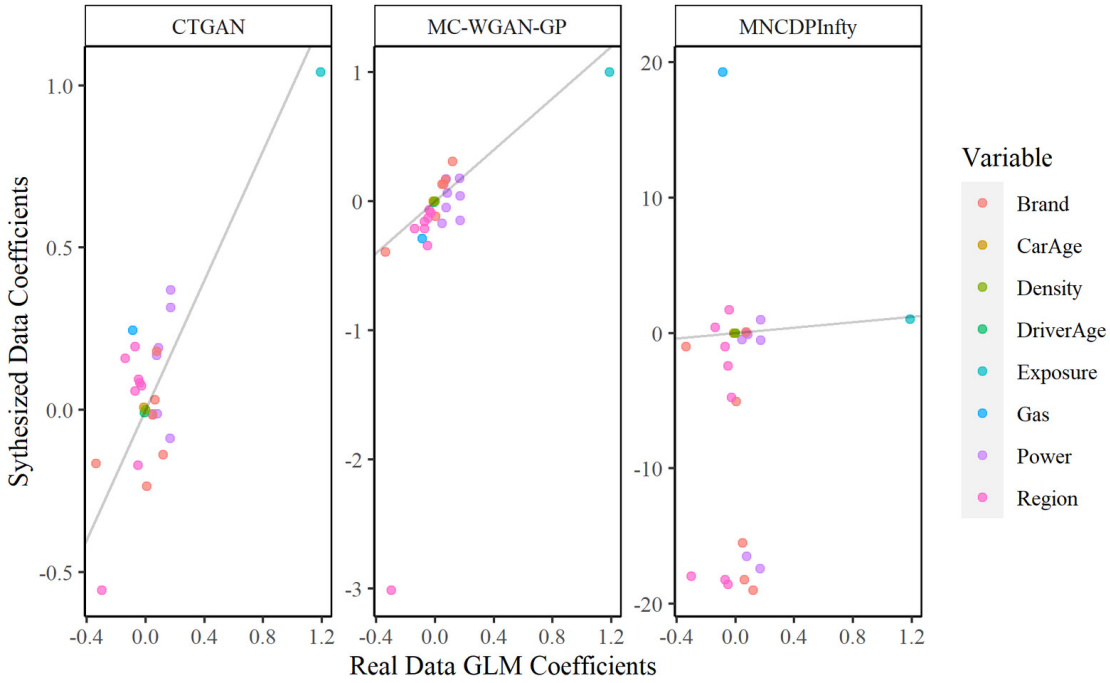


Figure 12. Comparison of the Poisson regression coefficients for the synthesized and real data.

Table 1. Poisson regression prediction errors (and bootstrapped 95% intervals) for the three main GANs.

Model	MAE (×1000)	MSE (×1000)
MC-WGAN-GP	5.0 (4.68, 5.44)	0.08 (0.06, 0.10)
CTGAN	10.8 (10.28, 11.32)	0.38 (0.36, 0.40)
MNCDPInfty	32.8 (32.38, 33.22)	3.36 (3.20, 3.52)

REFERENCES

Abadi, M., A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. 2016. "Deep Learning with Differential Privacy." In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 308–18. https://doi.org/10.1145/2976749.2978318.

Antonio, K., and R. Plat. 2014. "Micro-Level Stochastic Loss Reserving for General Insurance." *Scandinavian Actuarial Journal* 2014 (7): 649–69. https://doi.org/10.1080/03461238.2012.755938.

Arjovsky, M., S. Chintala, and L. Bottou. 2017. "Wasserstein Generative Adversarial Networks." *Proceedings of Machine Learning Research* 70:214–23. https://proceedings.mlr.press/v70/arjovsky17a.html.

Camino, R., C. Hammerschmidt, and R. State. 2018. "Generating Multi-Categorical Samples with Generative Adversarial Networks." Preprint, arXiv. https://doi.org/10.48550/arXiv.1807.01202.

Choi, E., S. Biswal, B. Malin, J. Duke, W. F. Stewart, and J. Sun. 2017. "Generating Multi-Label Discrete Patient Records Using Generative Adversarial Networks." *Proceedings of Machine Learning Research* 68:286–305. https://proceedings.mlr.press/v68/choi17a.html.

Dutang, C., and A. Charpentier. 2019. *CASdatasets: Insurance Datasets*. R package version 1.0-10.

Dwork, C., and A. Roth. 2014. "The Algorithmic Foundations of Differential Privacy." *Foundations and Trends in Theoretical Computer Science* 9 (3–4): 211–407. https://doi.org/10.1561/0400000042.

Frid-Adar, M., E. Klang, M. Amitai, J. Goldberger, and H. Greenspan. 2018. "Synthetic Data Augmentation Using GAN for Improved Liver Lesion Classification." In *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, 289–93. https://doi.org/10.1109/ISBI.2018.8363576.

Gabrielli, A., and M. V. Wüthrich. 2018. "An Individual Claims History Simulation Machine." *Risks* 6 (2): 29. https://doi.org/10.3390/risks6020029.

Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. 2014. "Generative Adversarial Nets." *Advances in Neural Information Processing Systems* 27:2672–80. https://proceedings.neurips.cc/paper_files/paper/2014/file/f033ed80deb0234979a61f95710dbe25-Paper.pdf.

Gulrajani, I., F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. 2017. "Improved Training of Wasserstein GANs." *Advances in Neural Information Processing Systems* 30:5767–77. https://proceedings.neurips.cc/paper_files/paper/2017/file/892c3b1c6dccd52936e27cbd0ff683d6-Paper.pdf.

Kuo, K. 2019. "Generative Synthesis of Insurance Datasets." *arXiv*. https://doi.org/10.48550/arXiv.1912.02423.

Lin, Z., A. Khetan, G. Fanti, and S. Oh. 2018. "PacGAN: The Power of Two Samples in Generative Adversarial Networks." *Advances in Neural Information Processing Systems* 31:1498–1507. https://proceedings.neurips.cc/paper_files/paper/2018/file/288cc0ff022877bd3df94bc9360b9c5d-Paper.pdf.

Papernot, N., M. Abadi, Ú. Erlingsson, I. Goodfellow, and K. Talwar. 2017. "Semi-Supervised Knowledge Transfer for Deep Learning from Private Training Data." Preprint, arXiv. https://doi.org/10.48550/arXiv.1610.05755.

Pigeon, M., K. Antonio, and M. Denuit. 2013. "Individual Loss Reserving with the Multivariate Skew Normal Framework." *ASTIN Bulletin* 43 (3): 399–428. https://doi.org/10.1017/asb.2013.20.

———. 2014. "Individual Loss Reserving Using Paid–Incurred Data." *Insurance: Mathematics and Economics* 58:121–31. https://doi.org/10.1016/j.insmatheco.2014.06.012.

Schelldorfer, J., and M. V. Wüthrich. 2019. "Nesting Classical Actuarial Models into Neural Networks." SSRN. https://doi.org/10.2139/ssrn.3320525.

Tantipongpipat, U., C. Waites, D. Boob, A. A. Siva, and R. Cummings. 2019. "Differentially Private Mixed-Type Data Generation for Unsupervised Learning." Preprint, arXiv. https://arxiv.org/abs/1912.03250.

Wüthrich, M. V. 2018a. "Machine Learning in Individual Claims Reserving." *Scandinavian Actuarial Journal* 2018 (6): 465–80. https://doi.org/10.1080/03461238.2018.1428681.

———. 2018b. "Neural Networks Applied to Chain-Ladder Reserving." *European Actuarial Journal* 8 (2): 407–36. https://doi.org/10.1007/s13385-018-0184-4.

Xu, D., S. Yuan, L. Zhang, and X. Wu. 2018. "FairGAN: Fairness-Aware Generative Adversarial Networks." In *2018 IEEE International Conference on Big Data (Big Data)*, 570–75. IEEE.

Xu, L., M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni. 2019. “Modeling Tabular Data Using Conditional GAN.” *Advances in Neural Information Processing Systems* 32:7333–43. https://proceedings.neurips.cc/paper_files/paper/2019/file/254ed7d2de3b23ab10936522dd547b78-Paper.pdf.

APPENDICES

Table A.1. Hyperparameter settings for the MC-WGAN-GP.

Hyperparameter	Explored Values	Chosen Value
loss_penalty	1, 5, 10, 20, 50	10
gen_bn_decay	0, 0.10, 0.25, 0.45, 0.50, 0.90	0.90
gen_L2_reg	0, 0.00001, 0.0001, 0.001, 0.01	0
disc_L2_reg	0, 0.00001, 0.0001, 0.001, 0.01	0
learning_rate	0.001, 0.005, 0.01	0.01

APPENDIX A. MC-WGAN-GP HYPERPARAMETER TUNING

In the MC-WGAN-GP model, we tuned the following hyperparameters:

- Loss penalty: loss_penalty
- Generator batch norm decay: gen_bn_decay
- Discriminator batch norm decay: disc_bn_decay
- Generator L2 regularization: gen_L2_reg
- Discriminator L2 regularization: disc_L2_reg
- Learning rate: learning_rate

We used a random search to explore the settings and decided on the values in Table A.1.

APPENDIX B. MNCDP-GAN METHODOLOGY

This appendix gives details on the preprocessing of the variables from the French motor third-party liability frequency dataset for the MNCDP-GAN in B.1. It then explains the training procedure in B.2, the hyperparameter optimization in B.3, and the final setting in B.4.

APPENDIX B.1. PREPROCESSING

Because the Exposure variable should be capped at 1, all 421 samples with Exposure greater than 1 were removed. The target ClaimNb was converted to a categorical variable, since there are only five possible values (0 to 4) and it is highly skewed toward 0. The adjusted dataset, used to train the models, contains 412,748 samples explained by four numerical (DriverAge, CarAge, Exposure, and Density) and five categorical (ClaimNb, Power, Brand, Gas, and Region) variables.

For the MNCDP-GAN experiments, we tested multiple configurations that differed on the types of those adjusted dataset variables. The first, *baseline*, used these data unaltered. In the *all-cat* configuration, both Exposure and Density were binned into categorical variables, as was also done for the MC-WGAN-GP. In the *bin* configuration, the four numeric variables were binned and treated as categorical. Expert insight was required to independently determine the binning size for each feature.

For DriverAge, we used 10 bins of increasing size (more bins at younger ages) to approximate a normal distribution. For CarAge, we used a single bin for the value 0, with the rest split using 10 quantiles. The resulting distribution was thus closer to the uniform distribution. For Exposure, 12 bins were chosen to reflect the 12 months in a year, except for the value 1, which had a bin of its own. The resulting distribution was skewed toward that last bin. Finally, for Density, binning was applied to its natural logarithm and based on the deciles, so that the resulting distribution was almost uniform.

APPENDIX B.2. TRAINING

For each configuration of the MNCDP-GAN, the autoencoder (AE) and the GAN were trained independently, one after the other, since the GAN requires the decoder. The AE training employed over 20,000 iterations, which was determined to be sufficient for convergence. Before feeding the preprocessed data to the network, we normalized the numerical features using min/max normalization and encoded the categorical variables using one-hot vectors. Unlike the more common method for encoding N categories in $N-1$ dimensions, we used one dimension per category. Otherwise, the GAN would never generate the category not having a dimension of its own. The AE used the binary cross entropy loss.

We conducted the GAN training over 2 million iterations. This value was empirically determined based on obtained results. However, because of the known difficulty of training GANs (e.g., no stability guarantees), it could be adjusted depending on the configuration and the hyperparameters. The generator and discriminator used the zero-sum objective function as proposed by Arjovsky, Chintala, and Bottou (2017), that is, Equation (1), for their learning. As recommended in this same paper, the discriminator was updated more often than the generator in order to train until optimality. We also applied the authors' recommendations to use a linear activation as the output layer of the discriminator and the RMSProp (root mean square propagation) optimizer for the GAN.

For both the AE and the GAN, the preprocessed dataset was split, two-thirds for training and one-third for validation. The networks' basic architecture (the number and sequence of layers) was not changed from Tantipongpipat et al. (2019). Unless stated otherwise, the activation functions used were always LeakyReLU (leaky rectified linear unit) with a negative slope of 0.2. Because they depend on the input size and some hyperparameters (such as latent dimensions), the layer sizes varied from configuration to configuration. We made the design choices of using the

Adam (adaptive moment estimation) optimizer with gradient penalty for the AE, using the absolute bounds to clip the values of the WGAN gradients, and opting for layer normalization over batch normalization for the generator, following the recommendations of Gulrajani et al. (2017).

The algorithm for the differentially private stochastic gradient descent (DP-SGD) came from Tantipongpipat et al. (2019). For the differential privacy aspect, we computed the L2 clipping norms of the gradients for the decoder and discriminator as recommended by Abadi et al. (2016). Finally, before training the models used to obtain results, we tuned the hyperparameters using a random search.

APPENDIX B.3. HYPERPARAMETER OPTIMIZATION

For the training, we tuned the hyperparameters of the AE and GAN in two stages. In a random search, each combination of hyperparameters tested was randomly selected from the chosen grid. The number of search iterations was set based on a time/resources compromise. The strategy was to run multiple searches instead of a single big search. Each time, the search space was narrowed for more fine-tuning. In all cases, we conducted tuning in a nonprivate way ($\epsilon = \infty$) to reduce computation time.

The AE hyperparameters tested were minibatch size, compression dimension, learning rate, β_1 and β_2 parameters of the Adam optimizer, and the L2 penalty of the weight decay for the optimizer. In the first experiments, these last three hyperparameters did not materially affect the training, so we left them at their usual default values (0.9, 0.999, and 0, respectively). To evaluate the performance of each combination, we saved and sorted the validation and training losses and chose the combination with the lowest final validation loss. We then conducted a regular training to confirm that the model was not overfitting.

The GAN hyperparameters tested were minibatch size, latent dimension of the generator, learning rate, number of iterations of the discriminator before updating the generator once, L2 penalty of the weight decay of the optimizer, and α smoothing constant of the RMSProp optimizer. Once again, these last two optimizer hyperparameters did not materially affect the results, so they were left at 0 and 0.99, respectively. The performance evaluation of the GAN was not as straightforward as that of the AE. The losses of both the discriminator and the generator were plotted. Over time, we found that the desired discriminator loss curve dropped rapidly to near 0 and then converged to that value over training iterations. For the generator, a loss oscillating rather slowly around 0 (going positive for many thousands of iterations and then going negative, and so on) appeared a good indicator of performance. Fortunately, these tendencies could be spotted after only a few tens of thousands of training iterations. Hence, to reduce computation time, we limited the number of iterations for each combination to 100,000.

Once we identified a couple of potentially good combinations in this manner, we conducted a full training with over 2 million iterations for each one. We then evaluated the generated samples from these trained models in respect to the univariate distributions of each variable (versus the real distributions). We also compared the predictions on the target ClaimNb of a random forest regressor and a random forest classifier between the generated and real samples, and kept the combination giving the best overall results. Because of the GANs' lack of stability (even for the same hyperparameters and configuration), this best combination of hyperparameters was trained at least two more times with different seeds for the random number generator. We saved the model of whichever run gave the best results and used it for the final results.

For both the AE and the GAN, for a given configuration, the minibatch size and learning rate hyperparameters had the greatest impact on the results. When training differentially private models, the values of the hyperparameters were the same as those of their corresponding nonprivate configuration. To guarantee the privacy, both the AE and the GAN were trained from scratch (i.e., their nonprivate counterparts were not used at any point).

APPENDIX B.4. CONFIGURATIONS

As stated previously, we tested different configurations to observe the impact of changing the types of some features. These configurations and their hyperparameter values are listed in Table B.1. Note that, except for the types of features, the *baseline* and *all_cat* configurations share the same hyperparameters, because using the values of the first on the second gave good results.

During the AE training, the learning rate was reduced by a factor 0.2 when the validation loss reached a plateau (tolerance of 1×10^{-4}) for 1,000 iterations (i.e., the patience). Its minimum value was limited to $1/100$ of the initial learning rate shown in Table B.1. We tested two other hyperparameter choices, which are not listed in Table B.1 because they did not improve the results. The first used a Kaiming uniform initialization of the weights for the AE and GAN instead of the default PyTorch initialization. The second used the Adam optimizer for the GAN instead of RMSProp. Both optimizers gave almost the same results when using the same random seed.

Table B.1. Hyperparameter values for the different configurations.

Hyperparameters		Configuration		
		baseline	all_cat	bin
Features	DriverAge	Numerical	Numerical	Categorical
	CarAge	Numerical	Numerical	Categorical
	Density	Numerical	Categorical	Categorical
	Exposure	Numerical	Categorical	Categorical
AE	l2 norm clip	0.022	0.022	0.022
	Minibatch size	64	64	128
	Compression dim	25	25	50
	Learning rate	0.01	0.01	0.01
	β_1 (Adam)	0.9	0.9	0.9
	β_2 (Adam)	0.999	0.999	0.999
	L2 penalty	0	0	0
GAN	L2 norm clip	0.027	0.027	0.027
	Clip value	0.01	0.01	0.01
	Minibatch size	128	128	128
	Latent dim	25	25	30
	Learning rate	4.5×10^{-5}	4.5×10^{-5}	3.9×10^{-5}
	Discriminator updates	10	10	5
	Alpha (RMSProp)	0.99	0.99	0.99
	L2 penalty	0	0	0